LPIC-303 - Topic 325: Cryptography

Bruno Ferreira - 2018 - 2019 - brunosilvaferreira@protonmail.com

325.1 X.509 Certificates and Public Key Infrastructures - 28 de Agosto 2018

X.509 - RFC 3820 (Proxy Certificate Profile) - Leitura da RFC

Caput - The term Proxy Certificate is used to describe a certificate that is derived from, and signed by, a normal X.509 Public Key End Entity Certificate or by another Proxy Certificate for the purpose of providing restricted proxying and delegation within a PKI based authentication system.

- Terminologia
    - CA - Certification Authority
    - EEC - End Entity Certificate - é a chave pública/Certificado emitido para uma entidade final, usuário/sistema/serviço, por uma CA.
    - PKC - Public key certificate - Mesma coisa da ECC. Sinônimos.
    - PC - Proxy Certificate - o perfil que é definido pela RFC 3820.
    - PI - Proxy Issuer - É a entidade com um EEC ou PC que emite um PC. O PC é assinado usando a chave privada associada com a chave pública dentro do certificado do PI.
    - AC - Attribute Certificate.
    - AA - Attribute Authority.
    - PEM - Privacy Enhanced Mail - Especifica a estrutura de codificação. Simplesmente indica o uso de base64 com cabeçalho e rodapé. O conteúdo do PEM está especificado no cabeçalho e rodapé. (Dar um vi em /etc/ssl/certs/qualquercoisa.pem)
    - DER - Distinguished Encoding Rules - Parecido com o PEM, foi criado para satisfazer os requisitos da especificação x.509 para transferência segura de dados. Conteúdo em binário. Por padrão o Windows usa o DER enquanto o

mundo Unix/Linux usa o formato PEM. Existe ferramentas como OpenSSL para fazer a conversão entre os 2.

- PKCS - Public Key Cryptography Standards - Padrão de segurança baseado no modelo RSA e no conceito de chave pública, sendo considerado o principal padrão para implementação de módulos de segurança baseados em criptografia assimétrica.
- CSR - Certificate Signing Request - é o que vc envia para uma terceira parte quando pede que um certificado seja assinado (pela terceira parte), a codificação pode ser PEM ou DER.
- .crt - Certificate - Usualmente um certificado X509v3, com codificação PEM ou DER, contendo uma chave pública, porém, contém muito mais informações como a assinatura da autoridade certificadora (CA).
- CRL - Certificate Revocation List - Lista para checagem de status de um certificado
- OCSP - Online Certificate Status Protocol - Serviço online para checagem

Resumo: Um PC é uma chave pública x.509 com as seguintes propriedades:

1. É assinado tanto por um x.509 EEC or por outro PC. Este EEC ou PC que assinou é referido como o PI.
2. Pode assinar apenas outro PC. Não pode assinar um EEC.
3. Tem seu próprio par de chaves pública e privada, distinto de qualquer EEC/PC.
4. Tem uma identidade derivada da identidade do ECC que assinou o PC. Quando um PC é usado para autenticação, pode herdar os direitos da ECC que assinou seu PC, sujeito as restrições que são colocadas naquele PC pelo ECC.
5. Apesar disso, esta identidade derivada da identidade do ECC, também é única. Isto permite que esta identidade seja usada para autorização como uma identidade independente.
6. Deve conter uma nova extensão x.509 para identificá-la como um PC e para colocar as policies no uso do PC.

O ciclo de vida - O processo de criação de um PC(Proxy Certificate):

- Geração de um novo par de chaves pública/privada.
- Este par de chaves é usado para criar um pedido para um PC que deverá estar em conformidade com o perfil descrito na RFC 3820.
- Um PC, assinado por uma chave privada do EEC ou por outro PC, é criado em resposta ao pedido anterior. Durante este processo, o pedido de PC é verificado para garantir que o PC requisitado é válido(ex. Que não é um EEC, os campos do PC estão corretamente configurados, etc.)

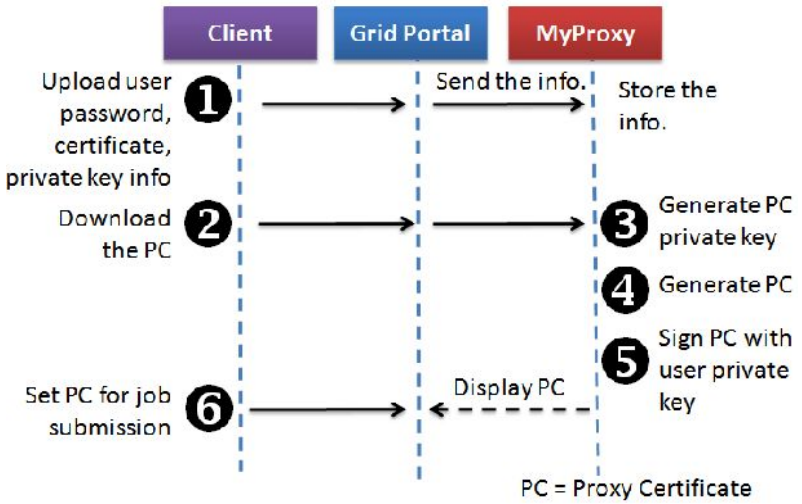


Figure 1: Issuance of Proxy Certificate with MyProxy

*Quando um PC é criado como parte de uma delegação de um entidade A para um entidade B, este processo é modificado pela atuação dos passos 1 e 2 dentro da entidade B, depois passando o pedido de PC da entidade B para a entidade A através de um canal autenticado e íntegro, depois a entidade A realiza o passo 3 e passa o PC de volta para a entidade B.

**Vantagem: Usando PC para SSO ajuda a fazer a autenticação via PKI x.509 mais fácil, permitindo usuários fazer “login” uma vez depois realizar várias operações de forma segura.

X.509 Certificate Fields - Campos e Extensões:

- Issuer - DEVE ser um ECC ou outro PC. Não pode ser Vazio
- Issuer Alternative Name - Não deve estar presente no PC
- Serial Number - Deve ser unico entre todos os PCs emitidos po um PI particular.
- Subject - O campo subject de um PC deve ser o campo Issue acrescido com um singular Common Name component(Esse singular CN pode ser o serial number para dar unicidade ao nome).
- Subject Alternative Name - Não deve estar presente no PC.
- Key usage and Extended Key Usage - Se o certificado do PI tem um Key Usage extension, o Digital Signature bit deve ser afirmado.
- Basic Constraints - O campo cA na extensão Basic Constraints não pode ser TRUE.
- ProxyCertInfo Extension - A presença dessa extensão indica que o certificado é um PC e independente do emissor do certificado tenha colocado alguma restrição no seu uso. Se o certificado é um PC, então a extensão proxyCertInfo deve estar presente, e esta extensão deve ser marcada como critical. Se um certificado não é um PC, então a extensão proxyCertInfo deve estar ausente. Esta extensão consiste de um campo obrigatório e dois opcionais: pCPathLenConstraint e proxyPolicy

Sobre a fase de Path Validation - Processamento básico de um PC:

1. Verifica informações básica do certificado. O certificado deve satisfazer as seguintes premissas:
    a. Se O certificado foi assinado com o working_public_key_algorithm using the working_public_key and the working_public_key_parameters
    b. A validade do certificado inclui a hora corrente.
    c. Se a variavel issuer name é igual ao working_issuer_name.
    d. Se o subject name do certificado é igual ao working_issuer_name com o componente CN acrescido.

*A aplicação PODE tomar decisões de autorização baseadas no subject distinguished name do PC ou em um dos PCs dentro da sua cadeia de emissão, ou em um ECC que serve como a raiz da cadeia.

Propagação da informação de autorização:

1. Pode estar incluída dentro de uma extensão da PKC ou em um AC separado. Porém usar um AC é a melhor opção para propagar a informação de autorização pois estará bindada a uma identidade.

Considerações de segurança:

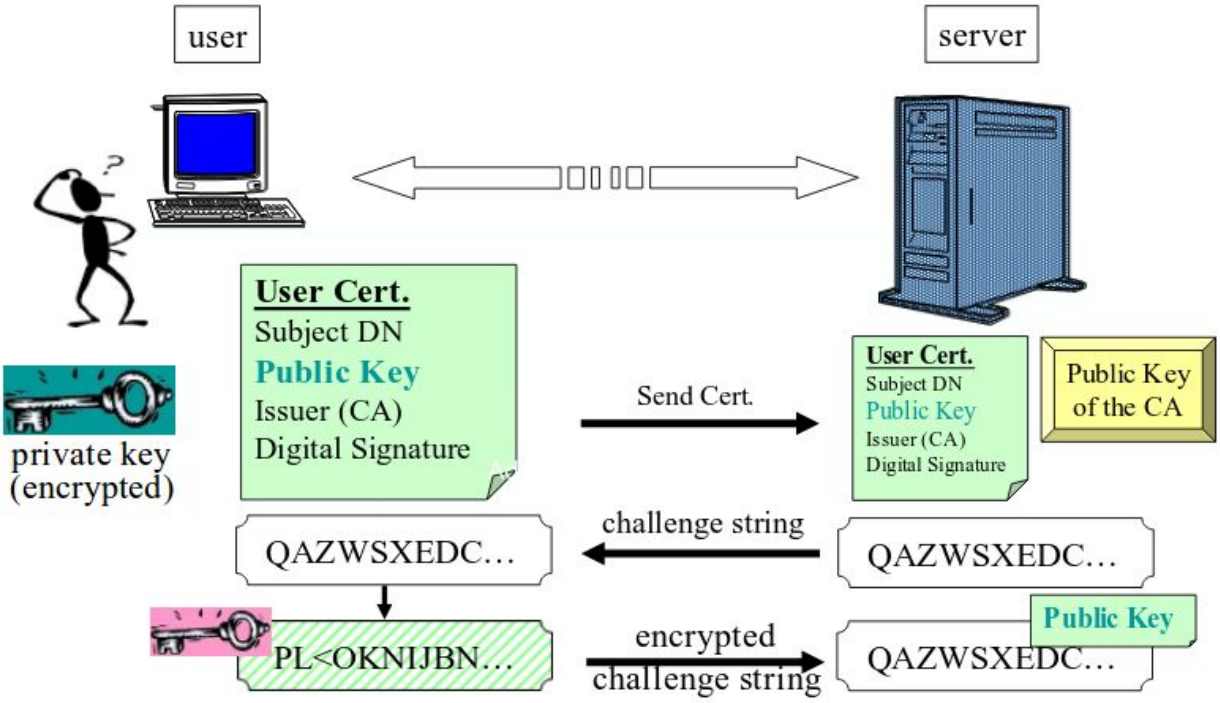
A Proxy Certificate is generally less secure than the EEC that issued it. This is due to the fact that the private key of a PC is generally not protected as rigorously as that of the EEC. For example, the private key of a PC is often protected using only file system security, in order to allow that PC to be used for single sign-on purposes. This makes the PC more susceptible to compromise.

In other words, the use of Proxy Certificates to provide single sign-on capabilities in an X.509 PKI environment can actually increase the security of the end entity certificates, because creation and use of the PCs for user authentication limits the exposure of the EEC private key to only the creation of the first level PC.

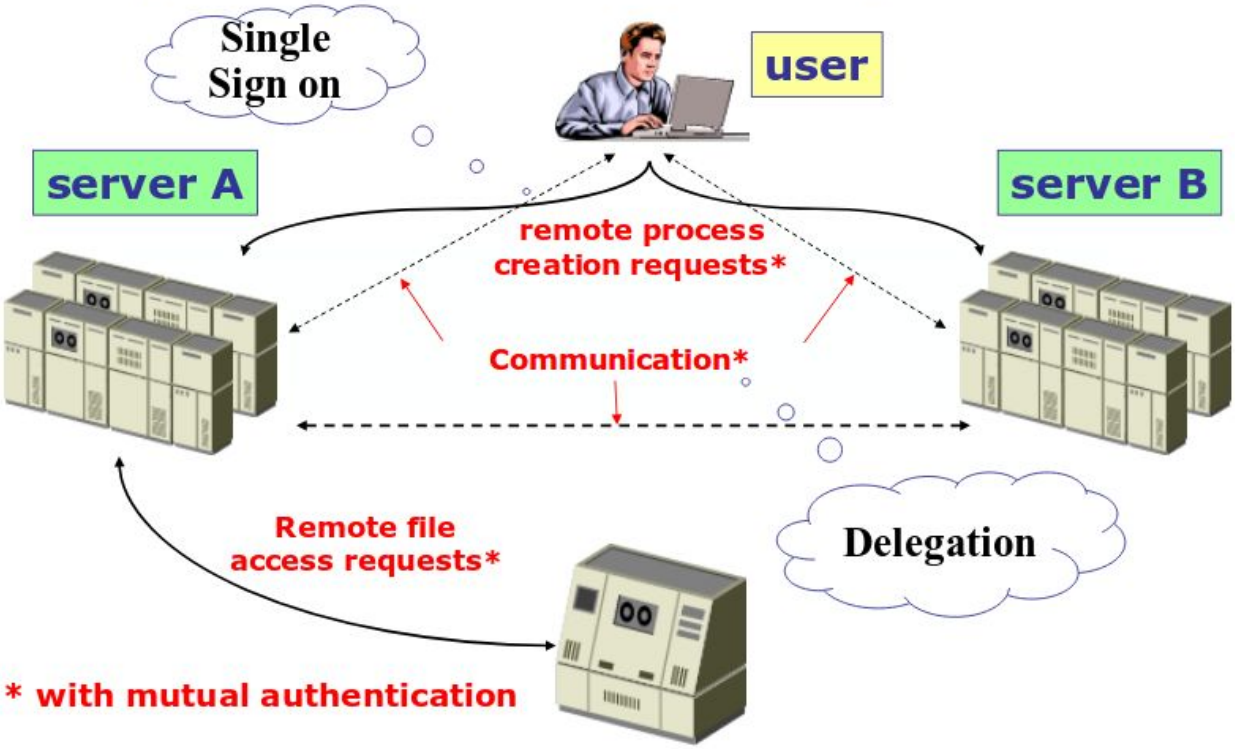
Para restringir um PC um EEC ou um PC pode limitar o que um novo PC pode ser usadado desligando os bit de Key Usage and Extended Key Usage.

O Protocolo Online Certificate Status (**OCSP**) é um protocolo de Internet utilizado para a obtenção do status de revogação de um X.509 certificado digital. É descrito no RFC 2560 e segue os padrões da Internet

How a user is authenticated by a server

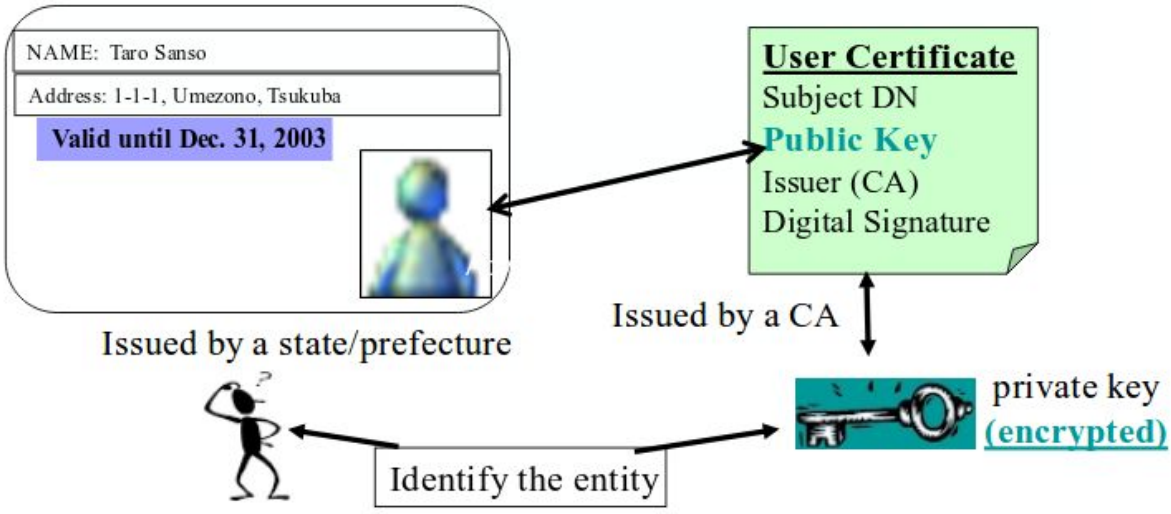


Requirements for Grid security



PKI and X.509 certificate (cont'd)

- X.509 certificates
  - Similar to a driving license. Photo on the license corresponds to a public key.
  - issued by a CA
  - Validity of the certificate depends on the opposite entity 's policy

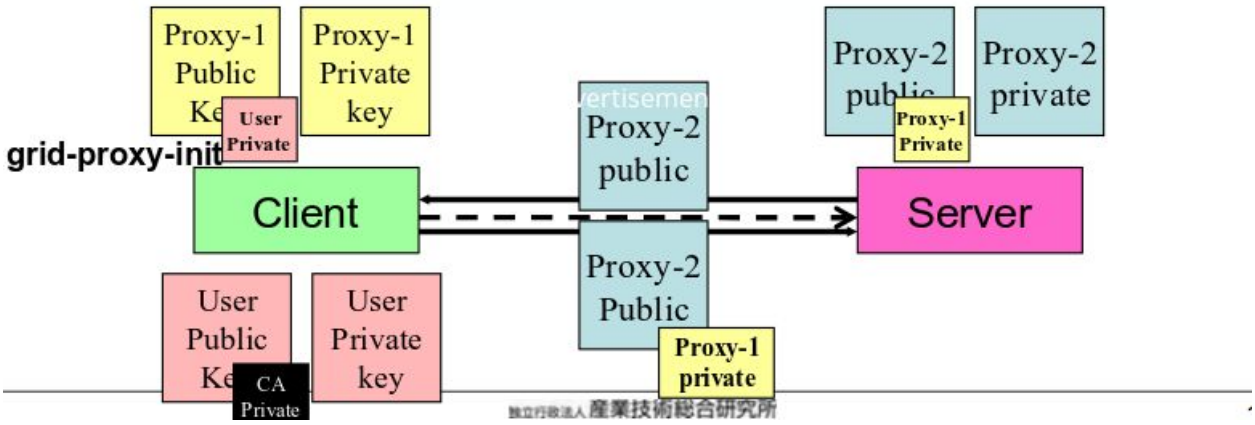


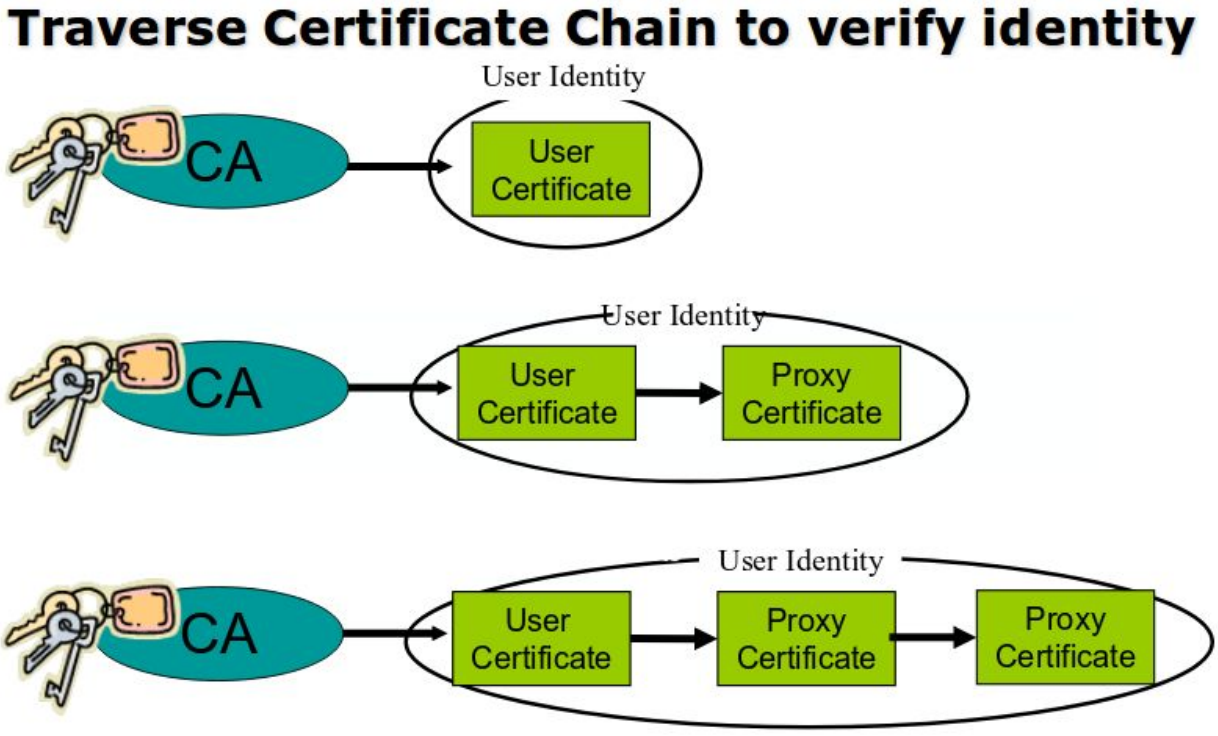
X.509 Proxy Certificate

- Defines how a short term, restricted credential can be created from a normal, long-term X.509 credential
  - A "proxy certificate" is a special type of X.509 certificate that is signed by the normal end entity cert, or by another proxy
  - Supports single sign-on & delegation through "impersonation"

Delegation

- Remote creation of a user proxy
- Results in a new private key and X.509 proxy certificate, signed by the original key
- Allows remote process to act on behalf of the user
- Avoids sending passwords or private keys across the network




Traverse Certificate Chain to verify identity
User Identity
CA
User Certificate
User Identity
CA
User Certificate
Proxy Certificate
User Identity
CA
User Certificate
Proxy Certificate
Proxy Certificate

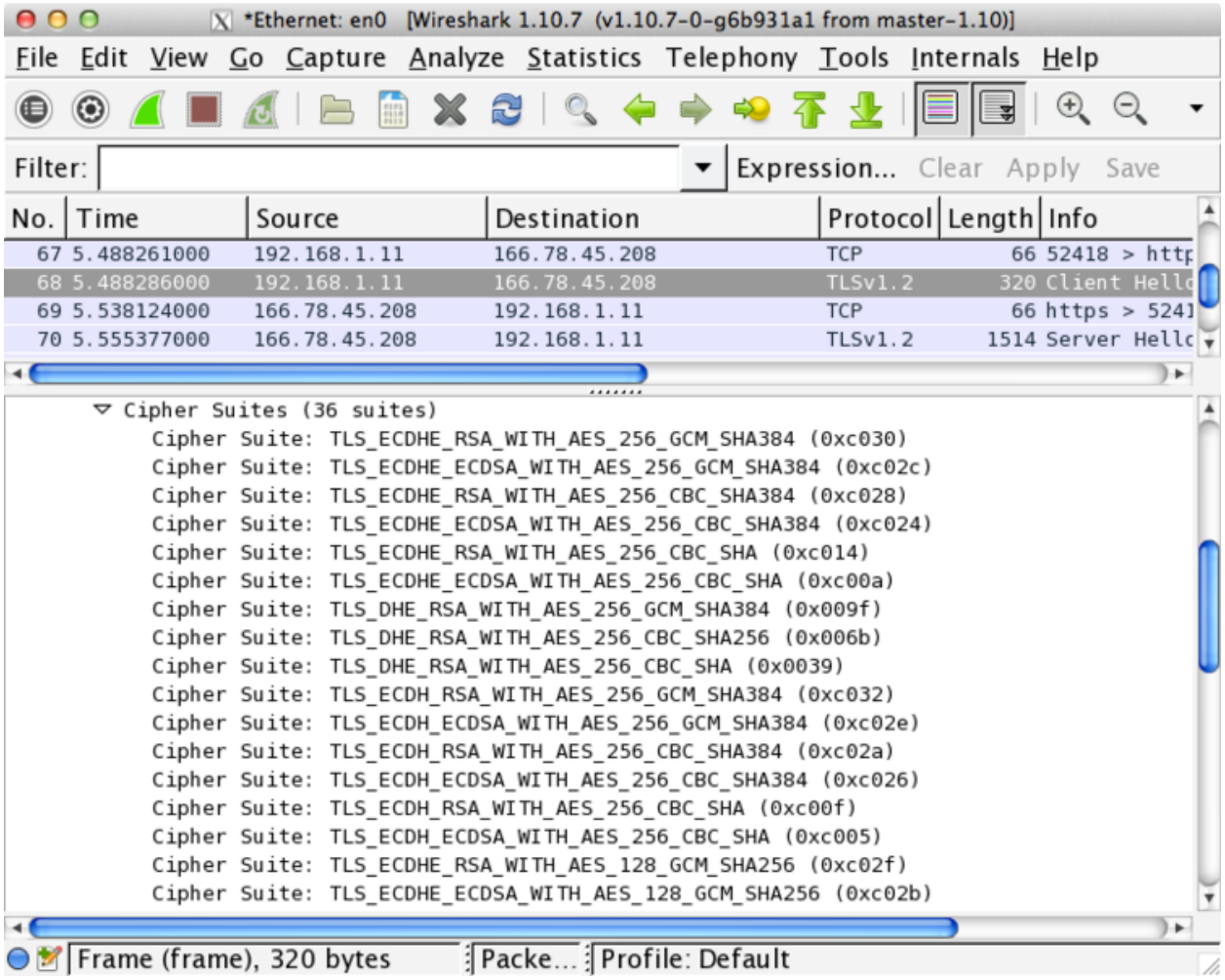
325.2 X.509 Certificates for Encryption, Signing and Authentication - 04 de Setembro de 2018

The goal of SSL was to provide secure communication using classical TCP sockets with very few changes in API usage of sockets to be able to leverage security on existing TCP socket code.

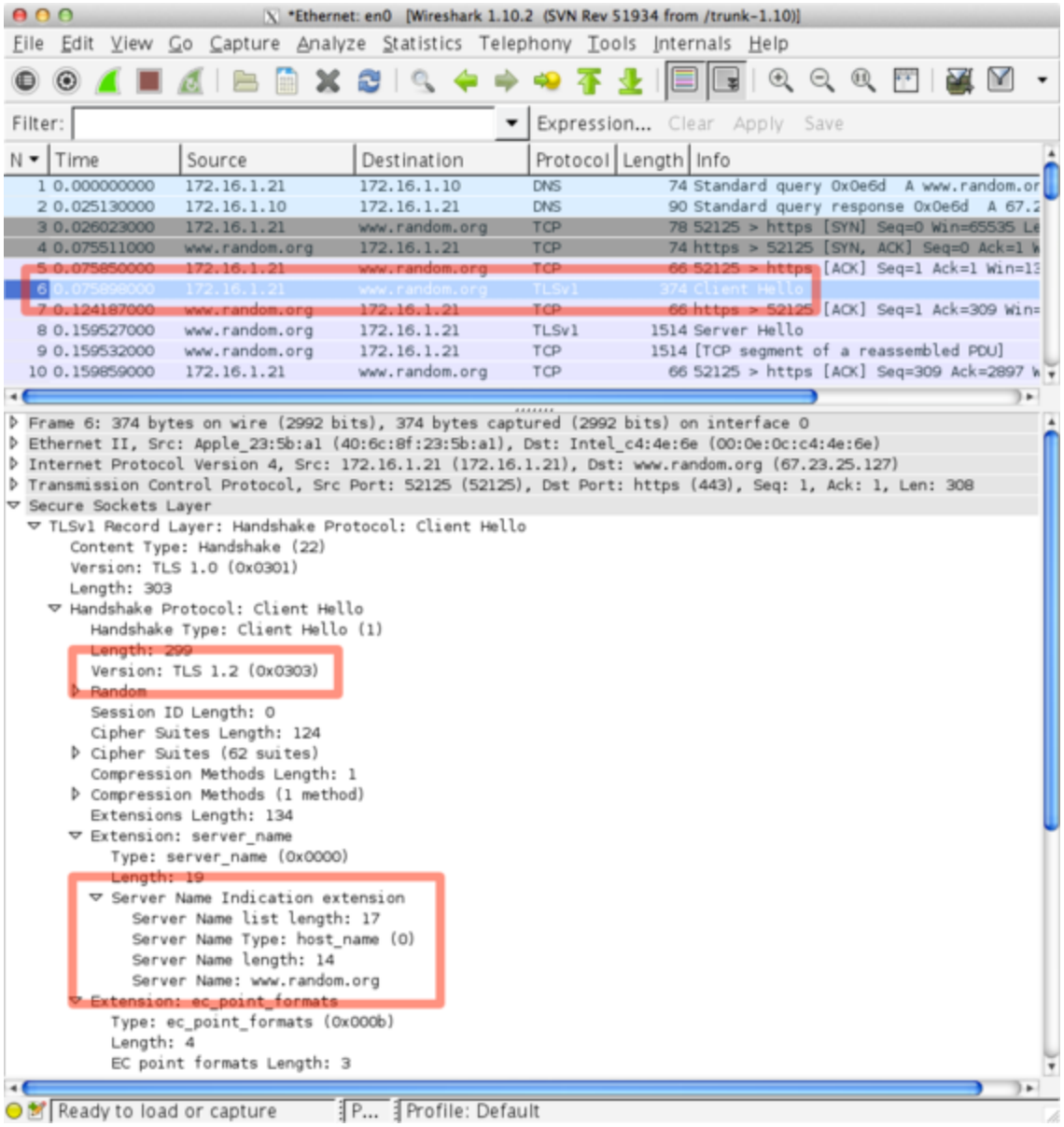
SSL3.0	TLS1.0	TLS1.1	TLS1.2
Separa os dados da camada de mensagem	Upgrade do SSL3.0	Abril de 2006 - RFC 4346	Ago - 2008 RFC 5246 - ficou mais flexível que o 1.1
128bits Message Auth Hash	Não é compatíuvel com SSL3.0	Proteção contra ataques CBC	Single hash. Elementos assinados digitalmente possuem um campo explicitando qual algoritmo usado.
Compatibilidade com SSL2.0	Requer suporte a DSS/DH		Varios requ

É recomendado usar o protocolo TLS e desabilitar totalmente o SSLv2/v3. E controlar opção de compatibilidade (protocol downgrade).

Nos protocolos SSL/TLS existe mais de 110 cifras disponíveis. No cliente hello é feita apresentação das cifras conhecidas e compatíveis para ele. Cabe ao servidor escolher uma. Negociação das cifras:



Extensões TLS - Server Name Indication - SNI - Permite que o cliente especifique logo no início do handshake qual nome de servidor que ele quer se conectar. Dessa forma o servidor pode saber qual certificado o cliente espera receber.



Criando certificado auto-assinado x509:

```
$cd /etc/apache2/; mkdir ssl; cd ssl/

$openssl req -x509 -nodes -days 365 -newkey rsa:2048 -out ./server.crt -keyout ./server.key
```

**server.crt - é o certificado em si / server.key a chave PRIVADA.*

Habilitando o SSL no Apache2:

```
$vi /etc/apache2/ports.conf; (Verificar se tem a linha Listen 443)

$a2enmod ssl
```

Para habilitar o SSL default basta criar um link simbólico:

```
$ln -s /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-enabled/000-default-ssl.conf
```

E editar as duas linhas:

```
$vi /etc/apache2/sites-enabled/000-default-ssl.conf

SSLCertificateFile /etc/apache2/ssl/server.crt

SSLCertificateKeyFile /etc/apache2/ssl/server.key

$service apache2 restart
```

(Por padrão o Apache2.4 do raspbian veio escutando somente IPV6, para forçar, basta mudar no ports.conf para **Listen 0.0.0.0:443(somente v4) ou Listen *:443(v4 e v6**)

Outro cenário:Criando certificado para ser assinado por uma CA:

```
$openssl req -newkey rsa:2048 -nodes -keyout server.key -out server.csr
```

HTST - HTTP Strict Transport Security

Resumo: Mecanismo de segurança para impedir ataques de downgrade de protocolo e sequestro de cookies. Is a web security policy mechanism that helps to protect websites against protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should interact with it using only secure HTTPS connections,[1] and never via the insecure HTTP protocol.

Funcionamento: Quando uma aplicação WEB informa para o user-agent que age em conformidade com a RFC6797, duas coisas acontecem: 1) Automaticamente transforma qualquer link inseguro http em links https. 2)

Proteção para os usuários da aplicação web: Contra alguns ataques passivos (eavesdropping), e ataques ativos de redes, MIT (SSL-stripping).

Arquivo conf(default-ssl.conf):

```
# HSTS (mod_headers is required) (15768000 seconds = 6 months)
Header always set Strict-Transport-Security "max-age=15768000"
```

Gerador de configuração para webserver seguro:
https://mozilla.github.io/server-side-tls/ssl-config-generator/

Fonte: https://www.smashingmagazine.com/2017/06/guide-switching-http-https/

Quadro resumo - Arquivos	
Arquivo	**Função**
server.key	Arquivo em formato PEM, contém a chave privada. (chmod 600)
server.pub	Arquivo em formato PEM, nem sempre é usado. Chave Púb.
server.csr	Arquivo em formato PEM, contendo as informações da organização, assim como a chave púb(server.pub), deve ser enviado para a autoridade certificadora para autorizar/assinar o certificado HTTPS.
server.crt	É o certificado HTTPS assinado pela autoridade certificadora. Formato PEM, incluindo a chave púb do server, informações organizacionais, assinatura da CA, validade e data de expiração, etc. O .crt não é um padrão, pode ser .cert ou .cer

Este processo é UNIVERSAL, será igual em cPanel, Linux, BSDs, Windows...

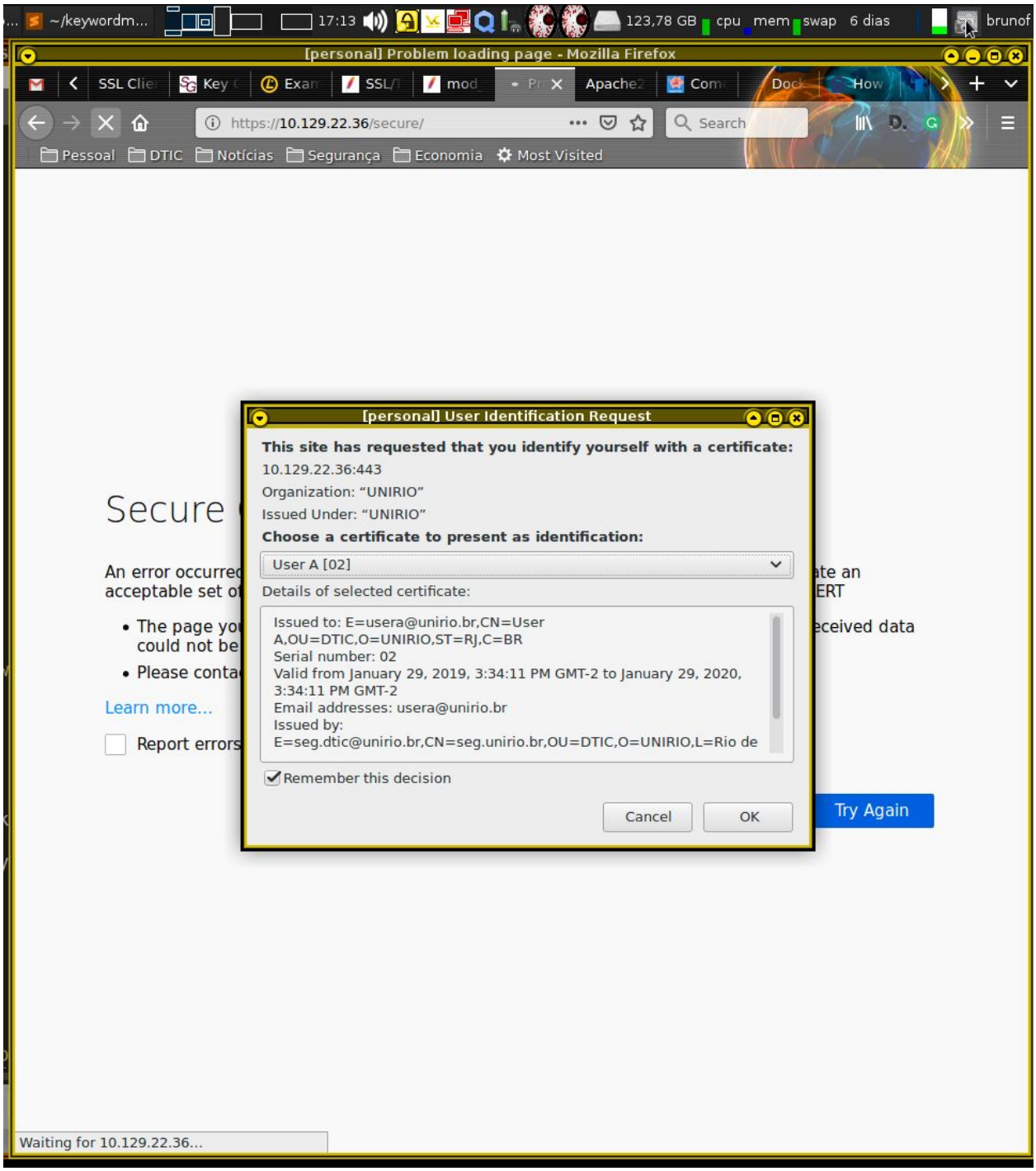
Quadro resumo - Tipos de Certificado HTTPS			
Tipo de certificado	**Domain validated(DV)**	**Organization validated(OV)**	**Extended validation(EV)**
	HTTPS	HTTPS Verified Legal owner	HTTPS Verified legal owner Owner info displayed in browser
Single domain	exemplo.com.br		
Multi domains	exemplo.com.br, www.exemplo.com.br, mail.exemplo.com.br, exemplo.br		
WildCard	*.exemplo.com.br (pega qualquer subdomínio)		N/A - Todos os nomes devem ser incluídos dentro do certificado e inspecionado pela CA.

*A Renovação é feita emitindo um novo certificado!

Ssl teste - https://github.com/drwetter/testssl.sh

Configure Apache HTTPD with mod_ssl to auth users using certificates

Resumo: Creating CA certificate; Create a Key and CSR for the Client; Sign the client certificate



325.3 Encrypted File System (Weight 3) - 7 de Fevereiro 2019

Understand block device and file system encryption

dm-crypt encrypts an entire block device with a single master key. An encrypted root filesystem makes tampering with the system far more difficult, as everything except the boot loader and (usually) the kernel is encrypted.

eCryptfs is another form of filesystem encryption on Linux; it encrypts a filesystem directory with some key or passphrase. eCryptfs sits on top of an existing filesystem. This makes eCryptfs an alternative choice if your filesystem or kernel does not support native filesystem encryption

Packages: use ecryptfs-utils for eCryptfs or cryptsetup for dm-crypt.

Block device vs stacked filesystem encryption

Aspect	Block device encryption	Stacked filesystem encryption
Encrypts	whole block devices	files
Container for encrypted data may be...	a disk or disk partition / a file acting as a virtual partition	a directory in an existing file system
Relation to filesystem	operates below filesystem layer: does not care whether the content of the encrypted block device is a filesystem, a partition table, a LVM setup, or anything else	adds an additional layer to an existing filesystem, to automatically encrypt/decrypt files whenever they are written/read
File metadata (number of files, dir structure, file sizes, permissions, mtimes, etc.) is encrypted	✔	✘ (file and dir names can be encrypted though)
Can be used to custom-encrypt whole hard drives (including partition tables)	✔	✘
Can be used to encrypt swap space	✔	✘
Can be used without pre-allocating a fixed amount of space for the encrypted data container	✘	✔
Can be used to protect existing filesystems without block device access, e.g. NFS or Samba shares, cloud storage, etc.	✘	✔
Allows offline file-based backups of encrypted files	✘	✔

Scenarios	Advantages	Disadvantages
#LUKS on a partition shows a basic and straight-forward set-up for a fully LUKS encrypted root.	• Simple partitioning and setup	• Inflexible; disk-space to be encrypted has to be pre-allocated
#LVM on LUKS achieves partitioning flexibility by using LVM inside a single LUKS encrypted partition.	• Simple partitioning with knowledge of LVM • Only one key required to unlock all volumes (e.g. easy resume-from-disk setup) • Volume layout not transparent when locked • Easiest method to allow **suspension to disk**	• LVM adds an additional mapping layer and hook • Less useful, if a singular volume should receive a separate key
#LUKS on LVM uses dm-crypt only after the LVM is setup.	• LVM can be used to have encrypted volumes span multiple disks • Easy mix of un-/encrypted volume groups	• Complex; changing volumes requires changing encryption mappers too • Volumes require individual keys • LVM layout is transparent when locked
#LUKS on software RAID uses dm-crypt only after RAID is setup.	• Analogous to LUKS on LVM	• Analogous to LUKS on LVM
#Plain dm-crypt uses dm-crypt plain mode, i.e. without a LUKS header and its options for multiple keys. This scenario also employs USB devices for `/boot` and key storage, which may be applied to the other scenarios.	• Data resilience for cases where a LUKS header may be damaged • Allows **Full Disk Encryption** • Helps addressing **problems** with SSDs	• High care to all encryption parameters is required • Single encryption key and no option to change it
#Encrypted boot partition (GRUB) shows how to encrypt the boot partition using the GRUB bootloader. This scenario also employs an EFI system partition, which may be applied to the other scenarios.	• Same advantages as the scenario the installation is based on (LVM on LUKS for this particular example) • Less data is left unencrypted, i.e. the boot loader and the EFI system partition, if present	• Same disadvantages as the scenario the installation is based on (LVM on LUKS for this particular example) • More complicated configuration • Not supported by other boot loaders
#Btrfs subvolumes with swap shows how to encrypt a **Btrfs** system, including the `/boot` directory, also adding a partition for swap, on UEFI hardware.	• Similar advantages as **#Encrypted boot partition (GRUB)** • Availability of Btrfs' features	• Similar disadvantages as **#Encrypted boot partition (GRUB)**

- **Use dm-crypt with LUKS to encrypt block devices**

The following commands create and mount the encrypted root partition (non boot).

```
# cryptsetup -y -v luksFormat --type luks2 /dev/sda2
# cryptsetup open /dev/sda2 cryptroot
# mkfs.ext4 /dev/mapper/cryptroot
# mount /dev/mapper/cryptroot /mnt
```

Check the mapping works as intended:

```
# umount /mnt
# cryptsetup close cryptroot
# cryptsetup open /dev/sda2 cryptroot
# mount /dev/mapper/cryptroot /mnt
```

- **Use eCryptfs to encrypt file systems, including home directories and Pam integration**

To set up a user account for full-home encryption, simply use:

```
ecryptfs-setup-private -u <your user name> -b
```

pam_ecryptfs is a PAM module that can use the login password to unwrap an ecryptfs mount passphrase stored in ~/.ecryptfs/wrapped-passphrase, and automatically mount a private cryptographic directory.

EXAMPLES

To unwrap a mount passphrase and automatically mount a private directory on login, add the following lines to

/etc/pam.d/common-auth:

```
    auth    required        pam_ecryptfs.so unwrap
```

/etc/pam.d/common-session:

```
session optional        pam_ecryptfs.so unwrap
```

Be aware of plain dm-crypt and EncFS

Contrary to LUKS, dm-crypt plain mode does not require a header on the encrypted device: this scenario exploits this feature to set up a system on an unpartitioned, encrypted disk that will be indistinguishable from a disk filled with random data, which could allow deniable encryption.

LUKS features like key management with multiple passphrases/key-files or re-encrypting a device in-place are unavailable with plain mode.

Plain dm-crypt encryption can be more resilient to damage than LUKS, because it does not rely on an encryption master-key which can be a single-point of failure if damaged. However, using plain mode also requires more manual configuration of encryption options to achieve the same cryptographic strength.

dm-crypt plain mode has parameters relating to how to locate the keyfile (e.g. --keyfile-size, --keyfile-offset). The dm-crypt LUKS mode does not need these, because each blockdevice contains a header with the cryptographic metadata at the beginning.

EncFS is a userspace stackable cryptographic file-system similar to eCryptfs, and aims to secure data with the minimum hassle.

Summary	Loop-AES	dm-crypt +/- LUKS	TrueCrypt	VeraCrypt	eCryptfs	EncFS
Encryption type	block device	block device	block device	block device	stacked filesystem	stacked filesystem
Note	longest-existing one; possibly the fastest; works on legacy systems	de-facto standard for block device encryption on Linux; very flexible	very portable, well-polished but abandoned	maintained fork of TrueCrypt	slightly faster than EncFS; individual encrypted files portable between systems	easiest one to use; supports non-root administration
Availability in Arch Linux	requires manually compiled, custom kernel	kernel modules: already shipped with default kernel; tools: device-mapper, cryptsetup	truecrypt	veracrypt	kernel module: already shipped with default kernel; tools: ecryptfs-utils	encfs
License	GPL	GPL	TrueCrypt License 3.1	Apache License 2.0; parts subject to TrueCrypt License v3.0	GPL	GPL
Encryption implemented in...	kernelspace	kernelspace	kernelspace	kernelspace	kernelspace	userspace (using FUSE)
Cryptographic metadata stored in...	?	with LUKS: LUKS Header	begin/end of (decrypted) device (format?)[dead link] 2018-07-15)	begin/end of (decrypted) device (format spec?)	header of each encrypted file	control file at the top level of each EncFs container
Wrapped encryption key stored in...	?	with LUKS: LUKS Header	begin/end of (decrypted) device (format spec?)[dead link] 2018-07-15)	begin/end of (decrypted) device (format spec?)	key file that can be stored anywhere	key file that can be stored anywhere [1][2]?

Usability features	Loop-AES	dm-crypt +/- LUKS	TrueCrypt	VeraCrypt	eCryptfs	EncFs
Non-root users can create/destroy containers for encrypted data	✘	✘	✘	✘	limited	✓
Provides a GUI	✘	✘	✓	✓	✘	✓ optional
Support for automounting on login	?	✓	✓ with systemd and /etc/crypttab	✓ with systemd and /etc/crypttab	✓	✓
Support for automatic unmounting in case of inactivity	?	?	?	?	?	✓
Security features	**Loop-AES**	**dm-crypt +/- LUKS**	**TrueCrypt**	**VeraCrypt**	**eCryptfs**	**EncFs**
Supported ciphers	AES	AES, Anubis, CAST5/6, Twofish, Serpent, Camellia, Blowfish,... (every cipher the kernel Crypto API offers)	AES, Twofish, Serpent	AES, Twofish, Serpent, Camellia, Kuznyechik	AES, Blowfish, Twofish...	AES, Blowfish, Twofish, and any other ciphers available on the system
Support for salting	?	✓ (with LUKS)	✓	✓	✓	?
Support for cascading multiple ciphers	?	Not in one device, but blockdevices can be cascaded	✓ AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES, Twofish-Serpent	✓ AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES, Twofish-Serpent	?	✘
Support for key-slot diffusion	?	✓ (with LUKS)	?	?	?	?
Protection against key scrubbing	✓	✓ (without LUKS)	?	?	?	?
Support for multiple (independently revocable) keys for the same encrypted data	?	✓ (with LUKS)	?	?	?	✘

Performance features	Loop-AES	dm-crypt +/- LUKS	TrueCrypt	VeraCrypt	eCryptfs	EncFs
Multithreading support	?	✓ [3][9]	✓	✓	?	?
Hardware-accelerated encryption support	✓	✓	✓	✓	✓	✓ [4][9]
Block device encryption specific	Loop-AES	dm-crypt +/- LUKS	TrueCrypt	VeraCrypt		
Support for (manually) resizing the encrypted block device in-place	?	✓	✗	✗		
Stacked filesystem encryption specific					eCryptfs	EncFs
Supported file systems					ext3, ext4, xfs (with caveats), jfs, nfs...	ext3, ext4, xfs (with caveats), jfs, nfs, cifs... [5][9]
Ability to encrypt filenames					✓	✓
Ability to not encrypt filenames					✓	✓
Optimized handling of sparse files					✗	✓
Compatibility & prevalence	Loop-AES	dm-crypt +/- LUKS	TrueCrypt	VeraCrypt	eCryptfs	EncFs
Supported Linux kernel versions	2.0 or newer	CBC-mode since 2.6.4, ESSIV 2.6.10, LRW 2.6.20, XTS 2.6.24	?	?	?	2.4 or newer
Encrypted data can also be accessed from Windows	?	?	✓	✓	?	✓ [6][9]
Encrypted data can also be accessed from Mac OS X	?	?	✓	✓	?	✓ [7][9]
Encrypted data can also be accessed from FreeBSD	?	?	✓ (with VeraCrypt)	✓	?	✓ [8][9]

cryptmount is a utility for GNU/Linux operating systems which allows an ordinary user to mount an encrypted filing system without requiring superuser privileges. It is aimed at Linux systems using the 2.6 kernel series or later. It handles both encrypted partitions as well as encrypted files.

dmsetup (part of the libdevmapper package) is a powerful tool for performing very low-level configuration of device-mapper targets, but requires root privileges and is not straightforward to use interactively for setting up dm-crypt targets.
cryptsetup (and the extended cryptsetup-luks) are valuable tools for performing lower-level configuration of the 'dm-crypt' target through libdevmapper. As yet, these are only directed at raw block devices, and would require separate configuration of loopback devices to handle filesystems stored in ordinary files. cryptsetup is supplied with scripts that can setup and mount encrypted filesystems at system startup, but this does not give ordinary users control over their own encrypted filesystems. (Support for mounting LUKS partitions via cryptmount is available in version 3.1)

sudo + dmsetup/cryptsetup - neither dmsetup nor cryptsetup is suitable for unrestricted use by ordinary users. 'sudo' would only appear to help get round the problem of needing root privileges if one had a script that performed the necessary calls to dmsetup/cryptset/losetup before mounting the filesystem. Such a script would probably have to perform many of the tasks that cryptmount already handles.

PAM-mount allows filing systems to be automatically mounted when a user logs in, but this makes it more difficult to decouple normal logins & passwords from occasional access to encrypted data having a separate password.

325.4 DNS and Cryptography (Weight 5) - 8 de Março de 2019

Candidates should have experience and knowledge of cryptography in the context of DNS and its implementation using BIND. The version of BIND covered is 9.7 or higher.

Understanding of DNSSEC and DANE - DNS-based Authentication of Named Entities (DANE) is an Internet security protocol to allow X.509 digital certificates, commonly used for Transport Layer Security (TLS), to be bound to domain names using Domain Name System Security Extensions (DNSSEC).[1]

Gerando as Chaves para a zona:

```
$dnssec-keygen -a RSASHA256 -b 1024 example.com
$dnssec-keygen -a RSASHA256 -b 2048 -f KSK example.com
```

Incluir no final do arquivo de zona:

```
zone "example.com" IN {
	type master;file "db/example.com.db";
	key-directory "keys/example.com";
	inline-signing yes;auto-dnssec maintain;
};
```

Reload bind:

```
#rndc reloadserver
reload successful
```